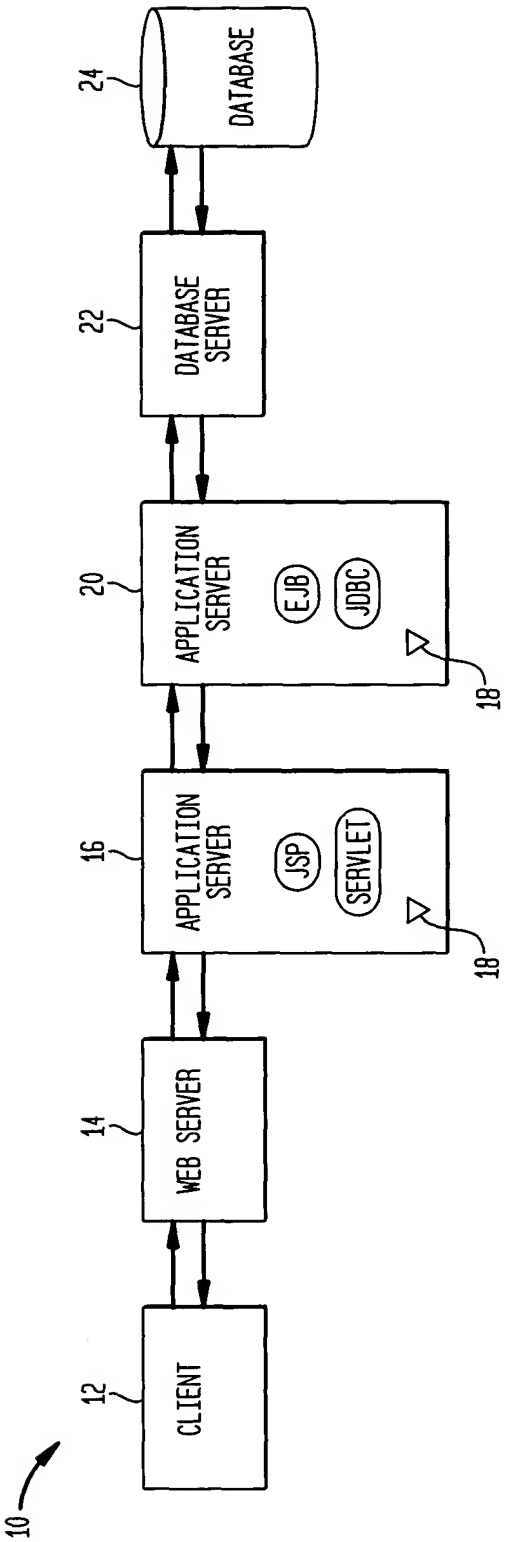

10
12
CLIENT
14
WEB SERVER
16
APPLICATION SERVER
JSP
SERVLET
18
20
APPLICATION SERVER
EJB
JDBC
18
22
DATABASE SERVER
24
DATABASE

FIG. 1

FIG. 2

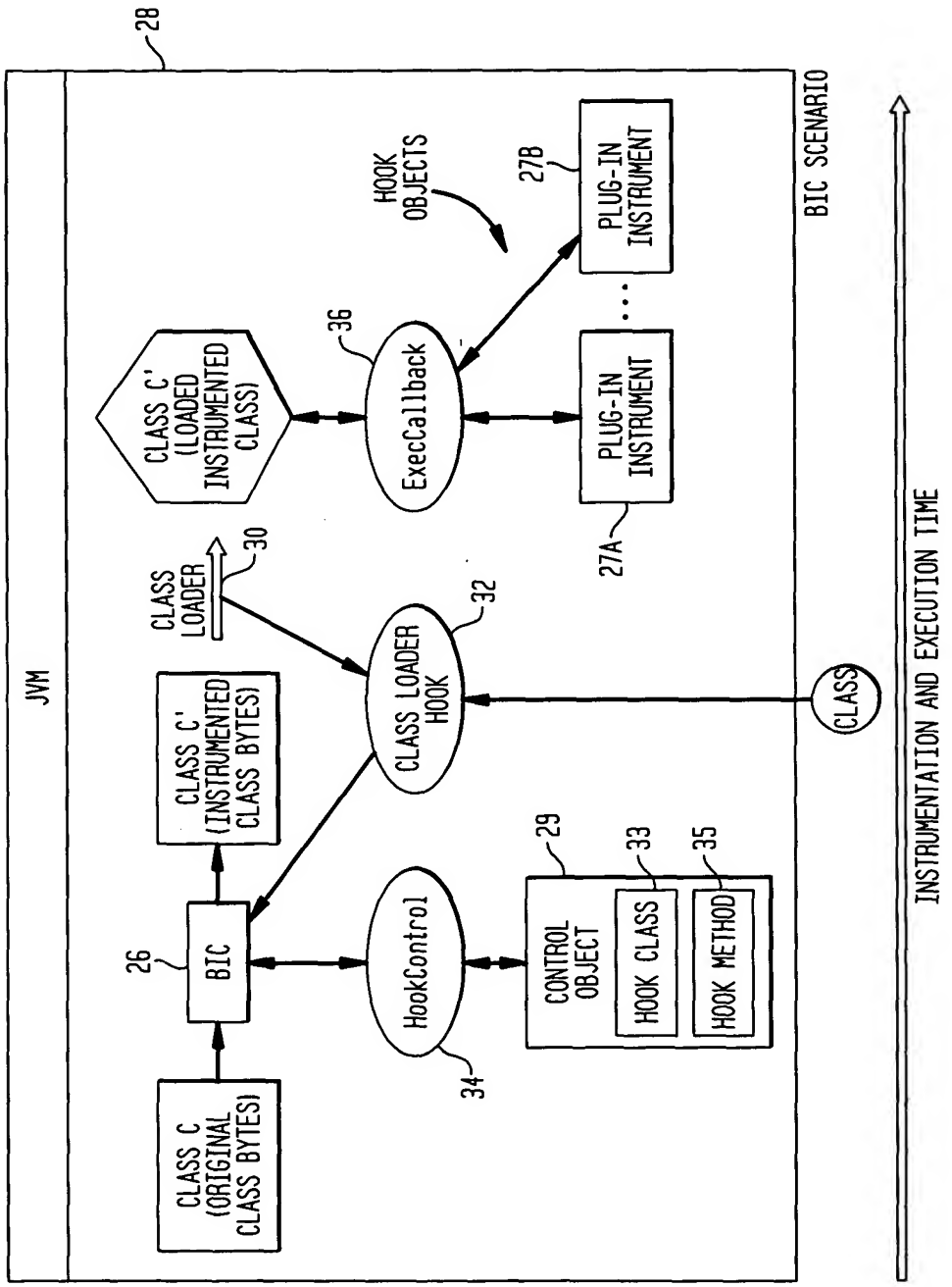

28
JVM
BIC SCENARIO
CLASS C (ORIGINAL CLASS BYTES)
26
BIC
CLASS C' (INSTRUMENTED CLASS BYTES)
CLASS LOADER
30
CLASS C' (LOADED INSTRUMENTED CLASS)
HookControl
34
CLASS LOADER HOOK
32
ExecCallback
36
HOOK OBJECTS
CONTROL OBJECT
29
HOOK CLASS
33
HOOK METHOD
35
PLUG-IN INSTRUMENT
27A
PLUG-IN INSTRUMENT
27B
CLASS
INSTRUMENTATION AND EXECUTION TIME

FIG. 3

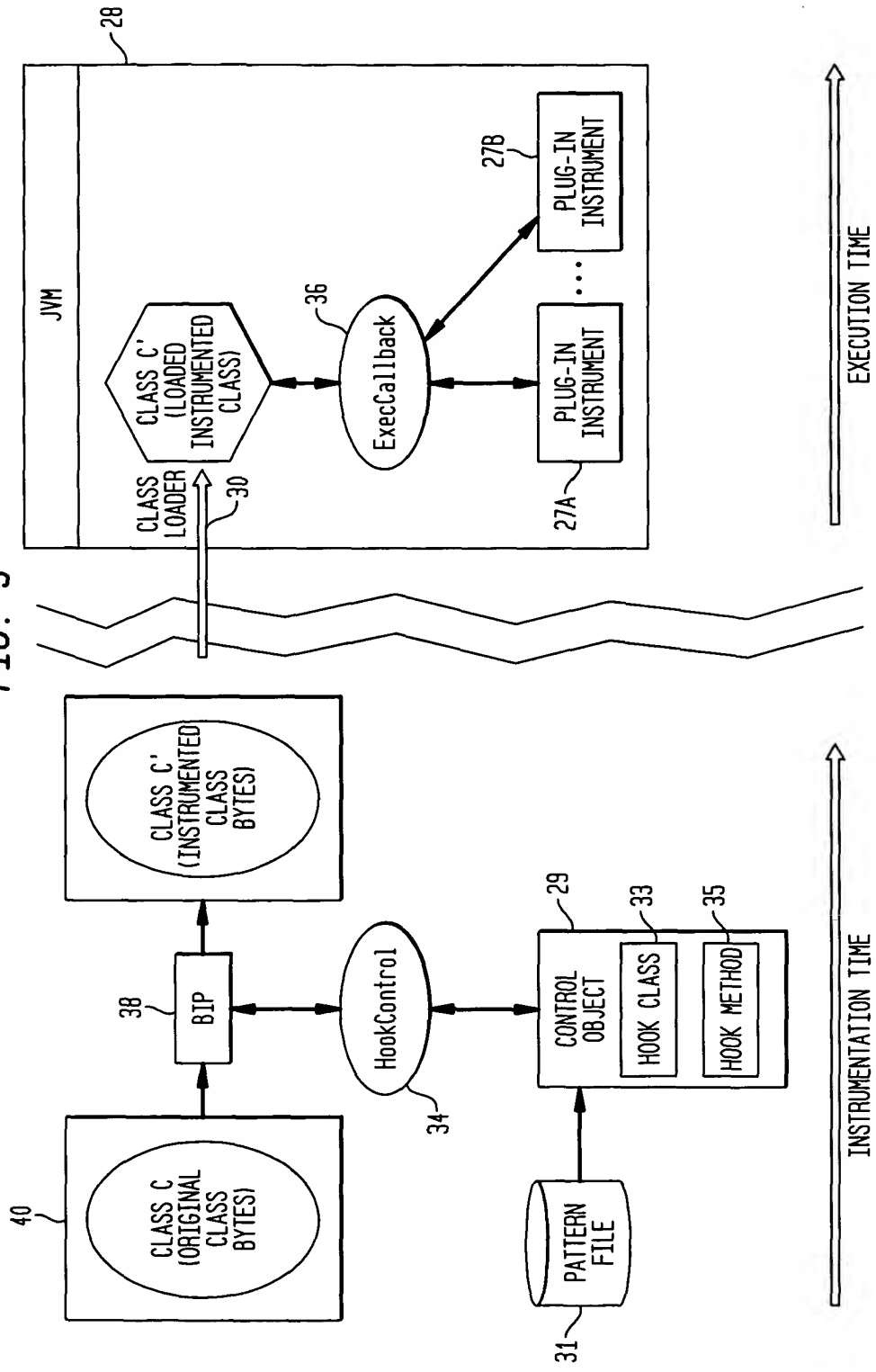

28
JVM
CLASS C' (LOADED INSTRUMENTED CLASS)
36
ExecCallback
27B
PLUG-IN INSTRUMENT
PLUG-IN INSTRUMENT
27A
CLASS LOADER
30
EXECUTION TIME
CLASS C' (INSTRUMENTED CLASS BYTES)
38
BIP
HookControl
34
29
CONTROL OBJECT
HOOK CLASS
33
HOOK METHOD
35
40
CLASS C (ORIGINAL CLASS BYTES)
PATTERN FILE
31
INSTRUMENTATION TIME

FIG. 4

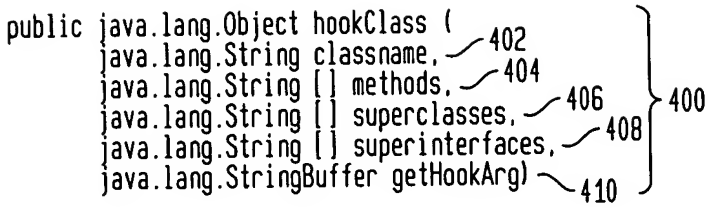

public java.lang.Object hookClass (
java.lang.String classname,
402
java.lang.String [] methods,
404
java.lang.String [] superclasses,
406
java.lang.String [] superinterfaces,
408
java.lang.StringBuffer getHookArg)
410
400

FIG. 5

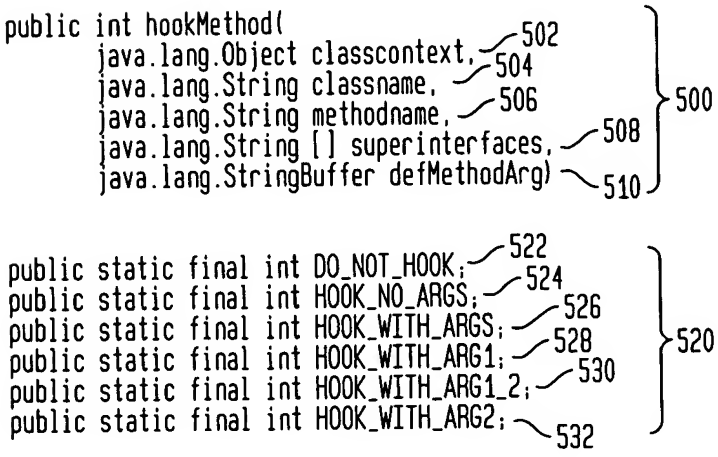

public int hookMethod(
java.lang.Object classcontext,
502
java.lang.String classname,
504
java.lang.String methodname,
506
java.lang.String [] superinterfaces,
508
java.lang.StringBuffer defMethodArg)
510
500
public static final int DO_NOT_HOOK;
522
public static final int HOOK_NO_ARGS;
524
public static final int HOOK_WITH_ARGS;
526
public static final int HOOK_WITH_ARG1;
528
public static final int HOOK_WITH_ARG1_2;
530
public static final int HOOK_WITH_ARG2;
532
520

FIG. 6A

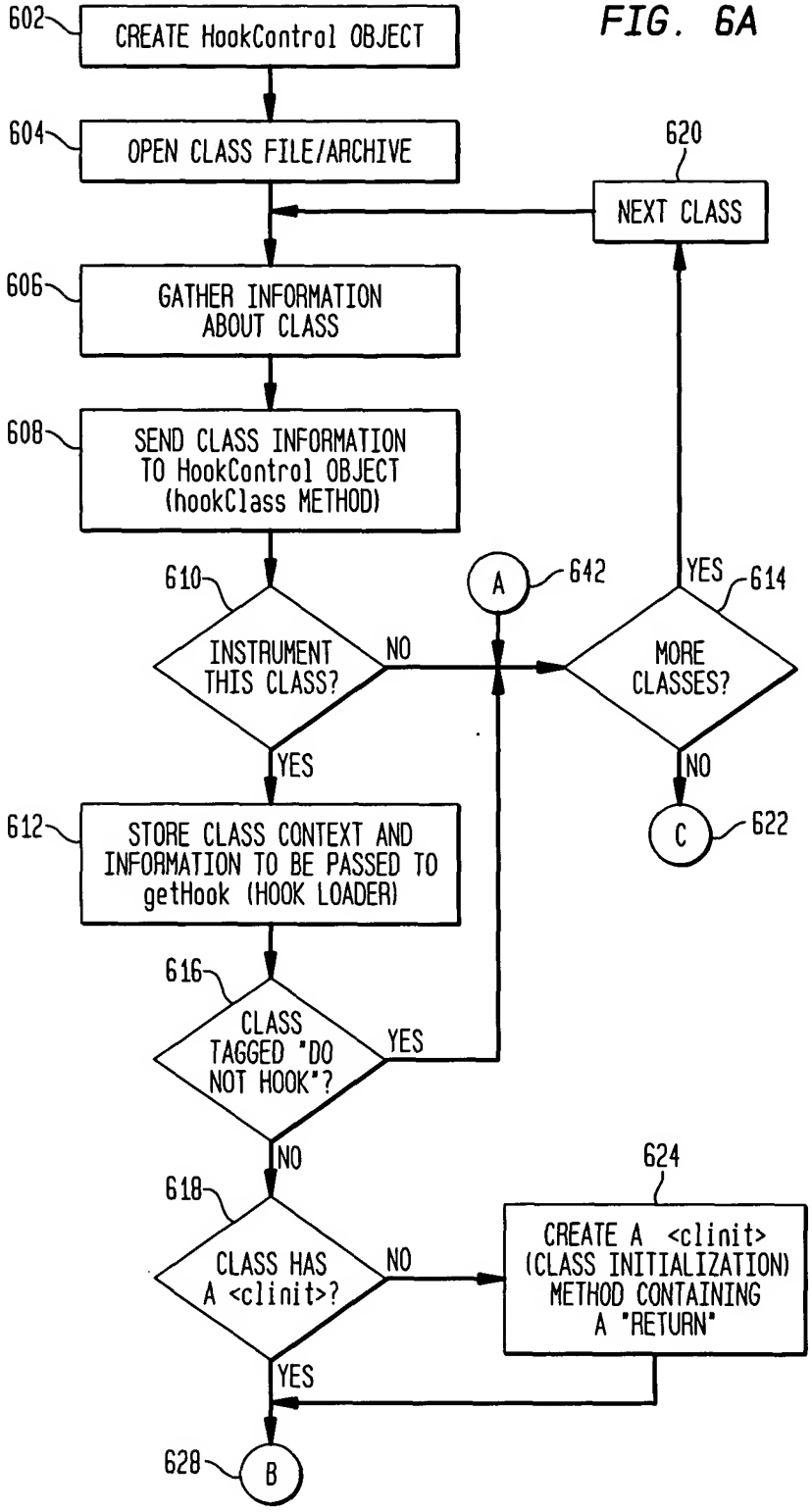

602
CREATE HookControl OBJECT
604
OPEN CLASS FILE/ARCHIVE
620
NEXT CLASS
606
GATHER INFORMATION ABOUT CLASS
608
SEND CLASS INFORMATION TO HookControl OBJECT (hookClass METHOD)
610
INSTRUMENT THIS CLASS?
NO
YES
642
A
614
MORE CLASSES?
YES
NO
622
C
612
STORE CLASS CONTEXT AND INFORMATION TO BE PASSED TO getHook (HOOK LOADER)
616
CLASS TAGGED "DO NOT HOOK"?
YES
NO
618
CLASS HAS A <clinit>?
NO
624
CREATE A <clinit> (CLASS INITIALIZATION) METHOD CONTAINING A "RETURN"
YES
628
B

FIG. 6B

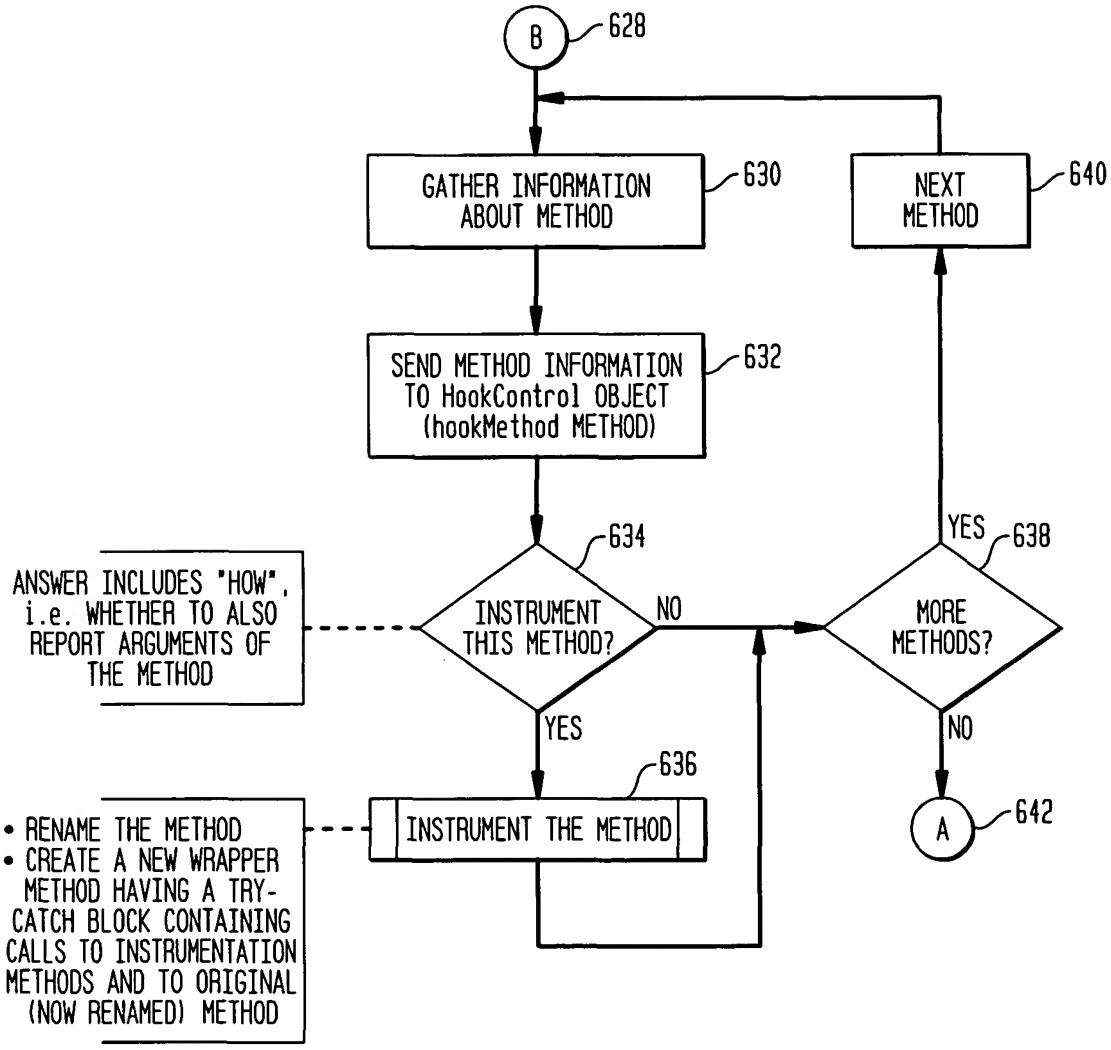

B
628
GATHER INFORMATION ABOUT METHOD
630
SEND METHOD INFORMATION TO HookControl OBJECT (hookMethod METHOD)
632
634
ANSWER INCLUDES "HOW", i.e. WHETHER TO ALSO REPORT ARGUMENTS OF THE METHOD
INSTRUMENT THIS METHOD?
NO
YES
636
• RENAME THE METHOD
• CREATE A NEW WRAPPER METHOD HAVING A TRY-CATCH BLOCK CONTAINING CALLS TO INSTRUMENTATION METHODS AND TO ORIGINAL (NOW RENAMED) METHOD
INSTRUMENT THE METHOD
NEXT METHOD
640
YES
638
MORE METHODS?
NO
A
642

FIG. 6C

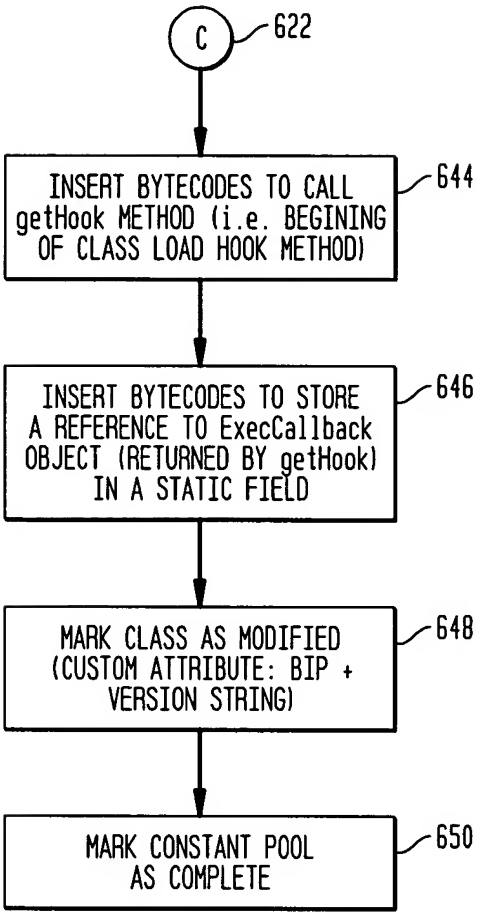

C
622
INSERT BYTECODES TO CALL getHook METHOD (i.e. BEGINING OF CLASS LOAD HOOK METHOD)
644
INSERT BYTECODES TO STORE A REFERENCE TO ExecCallback OBJECT (RETURNED BY getHook) IN A STATIC FIELD
646
MARK CLASS AS MODIFIED (CUSTOM ATTRIBUTE: BIP + VERSION STRING)
648
MARK CONSTANT POOL AS COMPLETE
650

FIG. 7

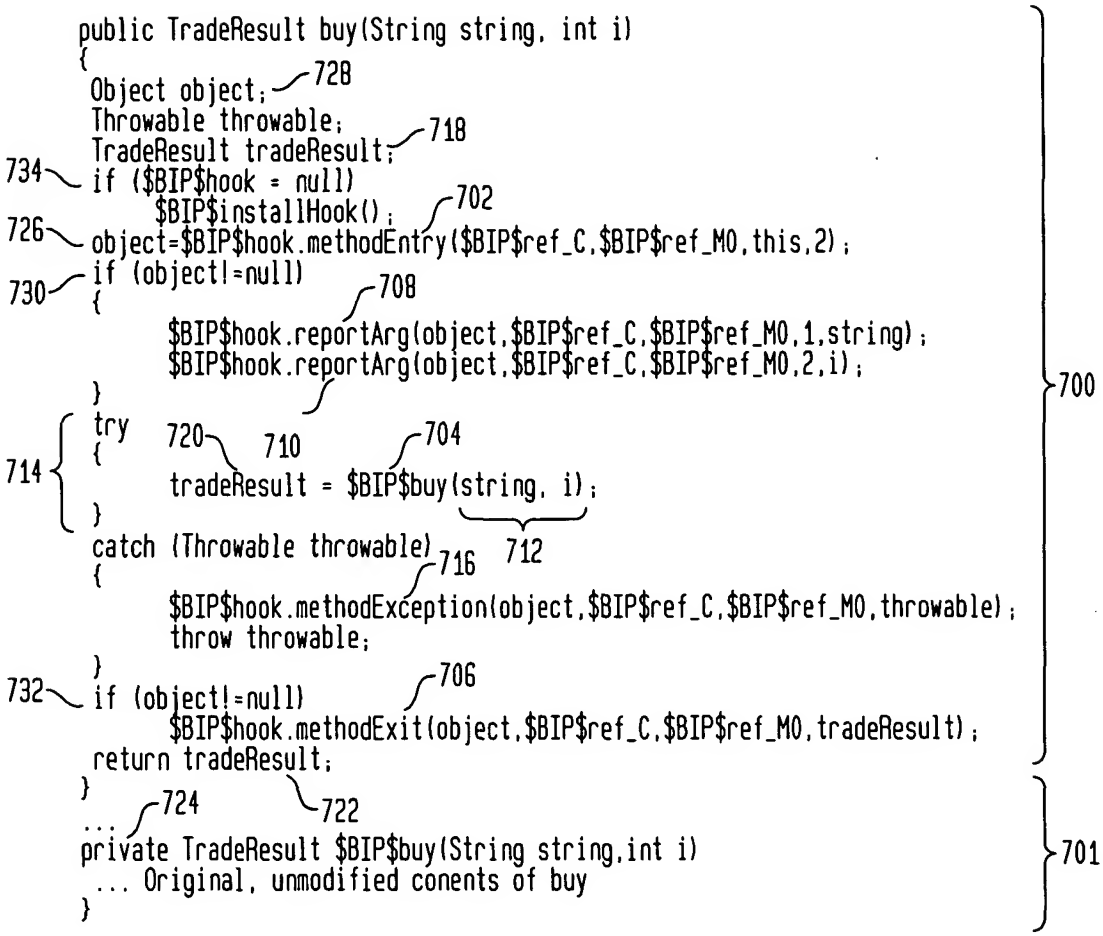

public TradeResult buy(String string, int i)
{
Object object;
728
Throwable throwable;
TradeResult tradeResult;
718
734
if (BIPhook = null)
BIPinstallHook();
702
726
object=BIPhook.methodEntry(BIPref_C,BIPref_M0,this,2);
730
if (object!=null)
{
708
BIPhook.reportArg(object,BIPref_C,BIPref_M0,1,string);
BIPhook.reportArg(object,BIPref_C,BIPref_M0,2,i);
}
700
try
720
710
704
714
{
tradeResult = BIPbuy(string, i);
}
712
catch (Throwable throwable)
716
{
BIPhook.methodException(object,BIPref_C,BIPref_M0,throwable);
throw throwable;
}
706
732
if (object!=null)
BIPhook.methodExit(object,BIPref_C,BIPref_M0,tradeResult);
return tradeResult;
}
724
722
...
701
private TradeResult BIPbuy(String string,int i)
... Original, unmodified conents of buy
}

FIG. 8A

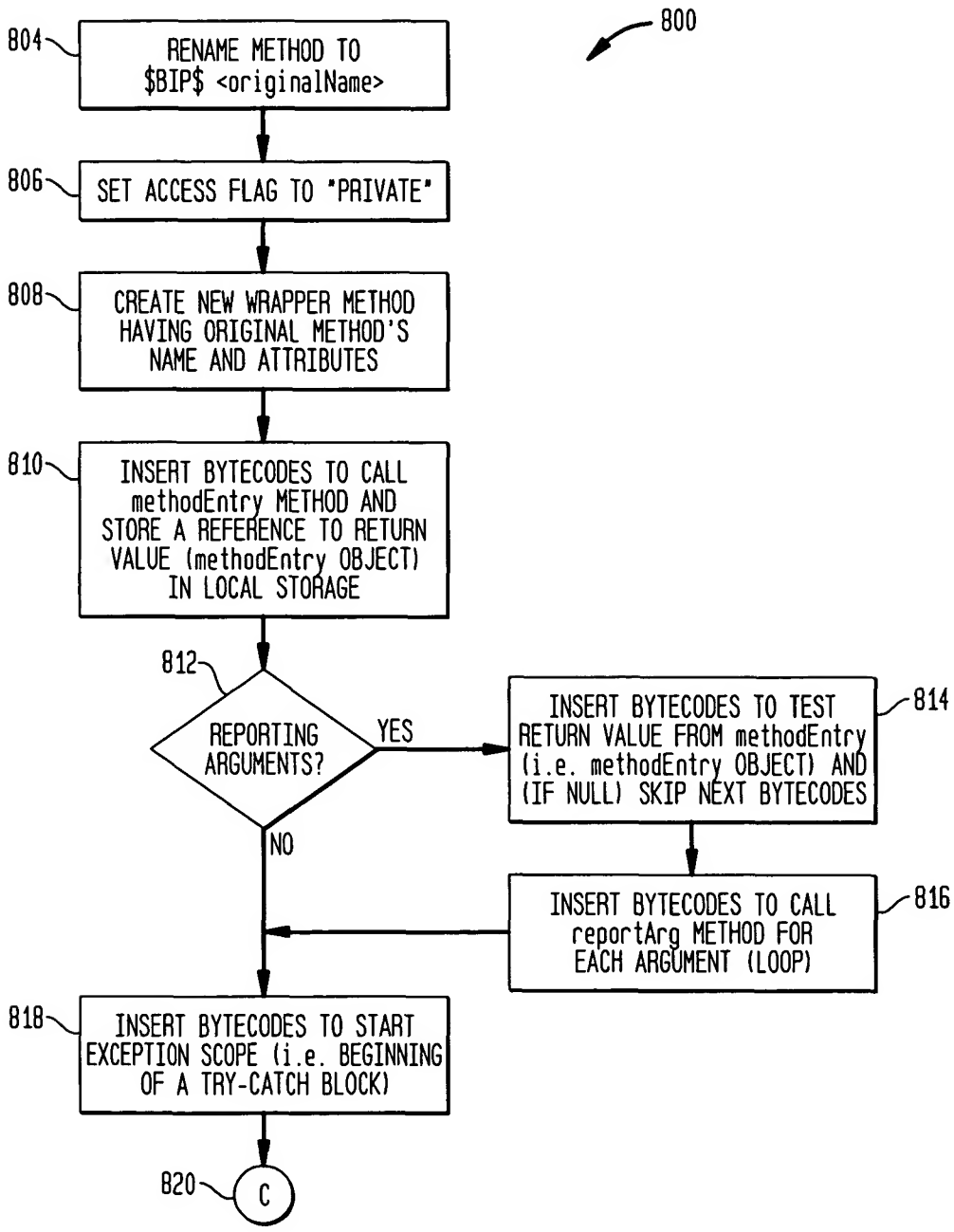

800
804
RENAME METHOD TO BIP <originalName>
806
SET ACCESS FLAG TO "PRIVATE"
808
CREATE NEW WRAPPER METHOD HAVING ORIGINAL METHOD'S NAME AND ATTRIBUTES
810
INSERT BYTECODES TO CALL methodEntry METHOD AND STORE A REFERENCE TO RETURN VALUE (methodEntry OBJECT) IN LOCAL STORAGE
812
REPORTING ARGUMENTS?
YES
814
INSERT BYTECODES TO TEST RETURN VALUE FROM methodEntry (i.e. methodEntry OBJECT) AND (IF NULL) SKIP NEXT BYTECODES
NO
816
INSERT BYTECODES TO CALL reportArg METHOD FOR EACH ARGUMENT (LOOP)
818
INSERT BYTECODES TO START EXCEPTION SCOPE (i.e. BEGINNING OF A TRY-CATCH BLOCK)
820
C

FIG. 8B

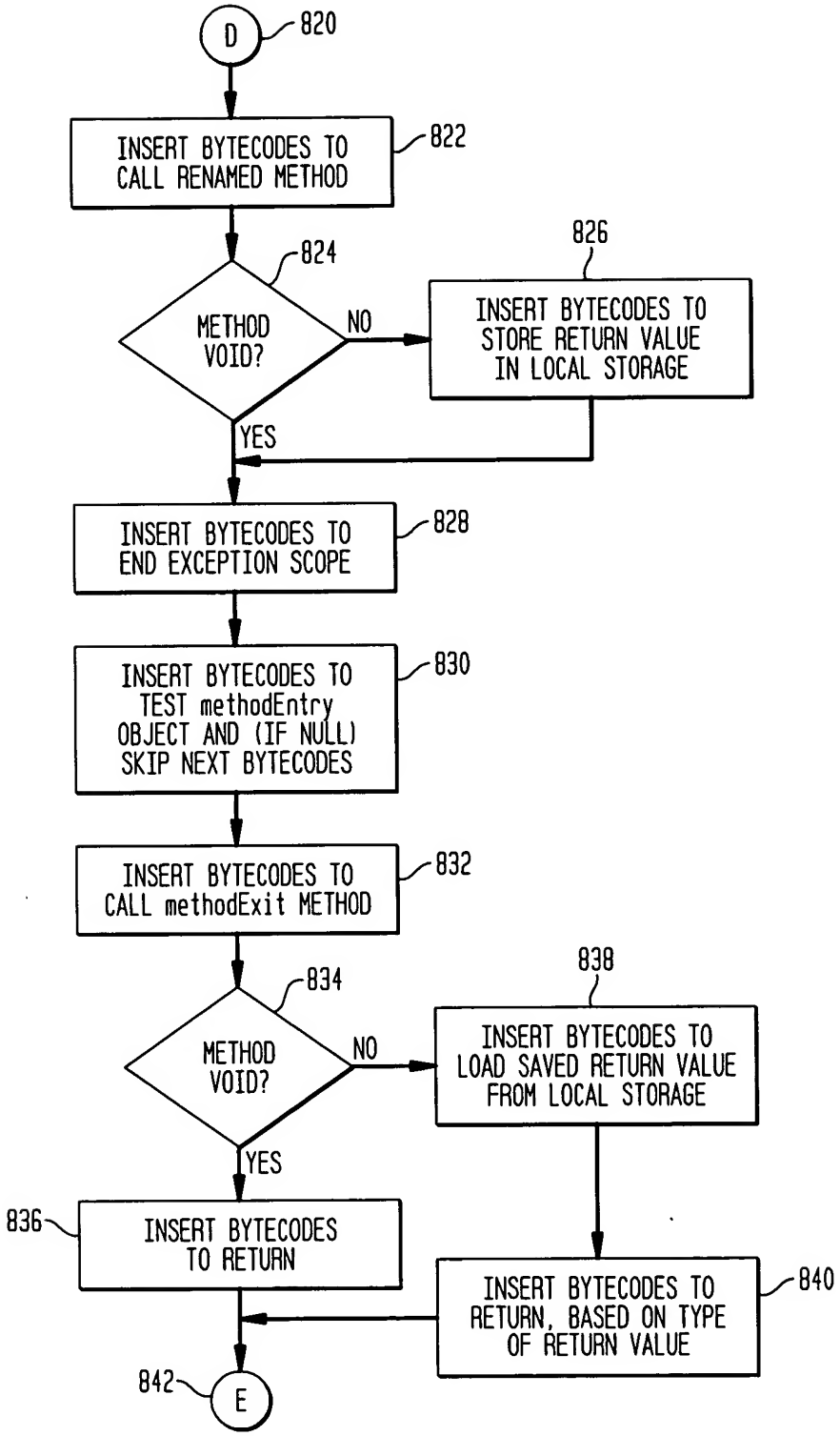

D
820
INSERT BYTECODES TO CALL RENAMED METHOD
822
824
METHOD VOID?
NO
826
INSERT BYTECODES TO STORE RETURN VALUE IN LOCAL STORAGE
YES
INSERT BYTECODES TO END EXCEPTION SCOPE
828
INSERT BYTECODES TO TEST methodEntry OBJECT AND (IF NULL) SKIP NEXT BYTECODES
830
INSERT BYTECODES TO CALL methodExit METHOD
832
834
METHOD VOID?
NO
838
INSERT BYTECODES TO LOAD SAVED RETURN VALUE FROM LOCAL STORAGE
YES
836
INSERT BYTECODES TO RETURN
INSERT BYTECODES TO RETURN, BASED ON TYPE OF RETURN VALUE
840
842
E

FIG. 8C

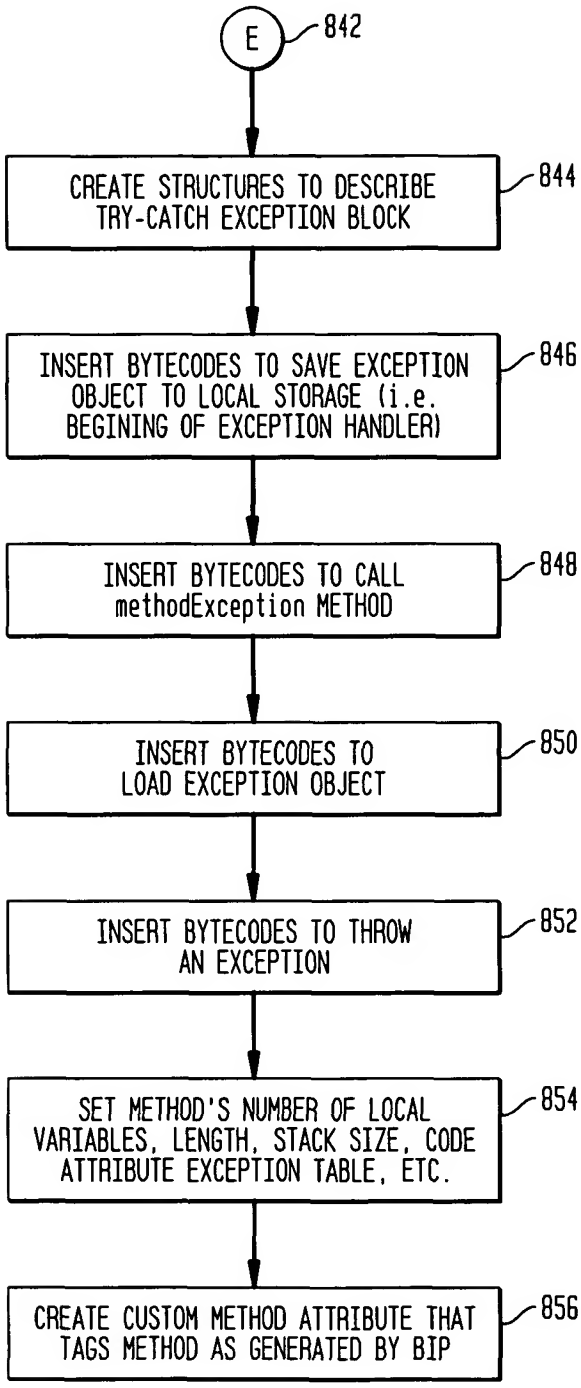

E
842
CREATE STRUCTURES TO DESCRIBE TRY-CATCH EXCEPTION BLOCK
844
INSERT BYTECODES TO SAVE EXCEPTION OBJECT TO LOCAL STORAGE (i.e. BEGINING OF EXCEPTION HANDLER)
846
INSERT BYTECODES TO CALL methodException METHOD
848
INSERT BYTECODES TO LOAD EXCEPTION OBJECT
850
INSERT BYTECODES TO THROW AN EXCEPTION
852
SET METHOD'S NUMBER OF LOCAL VARIABLES, LENGTH, STACK SIZE, CODE ATTRIBUTE EXCEPTION TABLE, ETC.
854
CREATE CUSTOM METHOD ATTRIBUTE THAT TAGS METHOD AS GENERATED BY BIP
856

FIG. 9

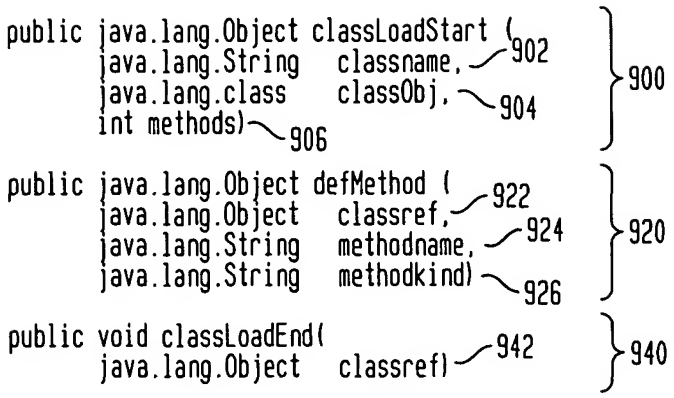

public java.lang.Object classLoadStart (
java.lang.String classname,
902
java.lang.class classObj,
904
int methods)
906
900
public java.lang.Object defMethod (
922
java.lang.Object classref,
924
java.lang.String methodname,
java.lang.String methodkind)
926
920
public void classLoadEnd(
java.lang.Object classref)
942
940

FIG. 10

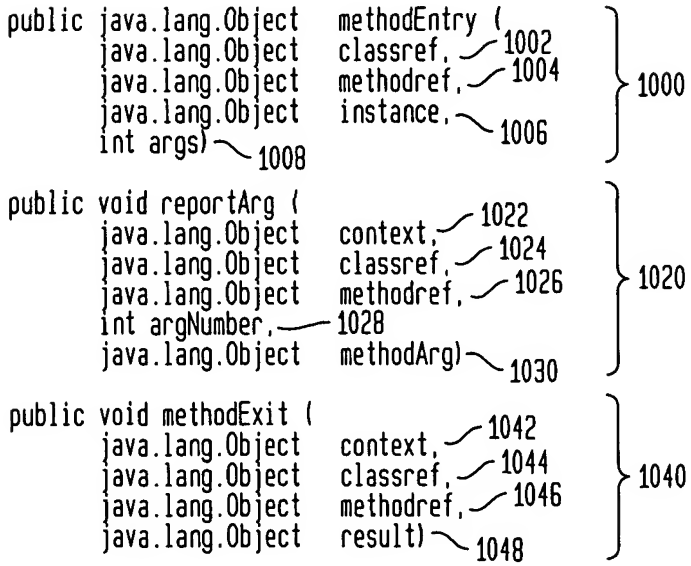

public java.lang.Object methodEntry (
java.lang.Object classref,
1002
java.lang.Object methodref,
1004
java.lang.Object instance,
1006
int args)
1008
1000
public void reportArg (
1022
java.lang.Object context,
1024
java.lang.Object classref,
1026
java.lang.Object methodref,
int argNumber,
1028
java.lang.Object methodArg)
1030
1020
public void methodExit (
1042
java.lang.Object context,
1044
java.lang.Object classref,
1046
java.lang.Object methodref,
java.lang.Object result)
1048
1040

FIG. 11

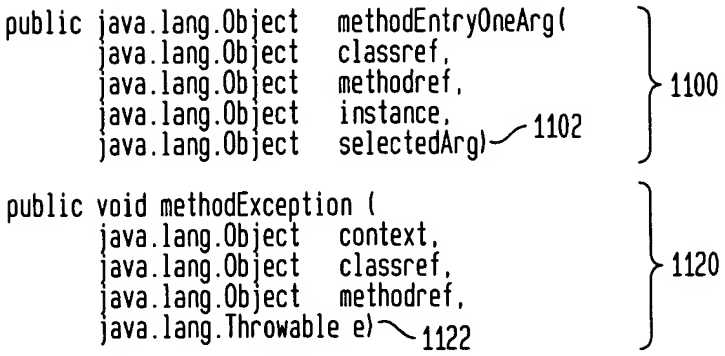

public java.lang.Object methodEntryOneArg(
java.lang.Object classref,
java.lang.Object methodref,
java.lang.Object instance,
java.lang.Object selectedArg)
1102
1100
public void methodException (
java.lang.Object context,
java.lang.Object classref,
java.lang.Object methodref,
java.lang.Throwable e)
1122
1120

FIG. 12

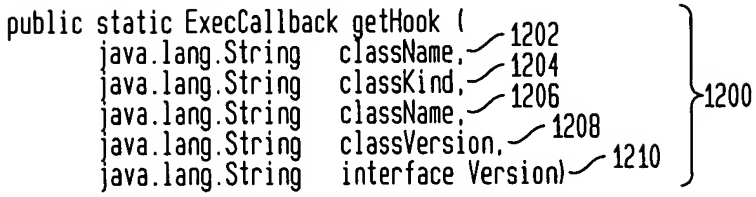

public static ExecCallback getHook (
java.lang.String className,
1202
java.lang.String classKind,
1204
java.lang.String className,
1206
java.lang.String classVersion,
1208
java.lang.String interface Version)
1210
1200

FIG. 13A

1300

```
// $Source: /data1/nebula/ccm/jade/ccm/import/arra_jlink/i2/bip/hook/RCS/NullExec?Callback.java,v $
// $Revision: 1.8 $ $Date: 2001/08/28 14:56:29 $ $Author: arav $
package i2.bip.hook;
/** An implementation of the ExecCallback that does nothing.
 * A suitable base class for a custom hook class.
 */
public class NullExecCallback
    // Explicit DoNotHook for BIC testing
    implements ExecCallback, DoNotHook {

    // Called at start of class initialization
    // Returns opaque class ref
    public Object classLoadStart(String classname, Class classObj, int methods)
    {
      return null;
    }

    // Called once for each instrumented method in the class.
    // Returns opaque method ref
    public Object defMethod(
      Object classref,
      String methodname,
      String methodkind)
    {
      return null;
    }

    // End of class initialization instrumentation
    public void classLoadEnd(Object classref) { }

    // Called at instrumented method entry.
    public Object methodEntry(
      Object classref,
      Object methodref,
      Object instance,
      int args)
    {
      return null;    // Disables methodExit & reportArg instrumentation
    }

    // Called at instrumented method entry when single arg requested.
```

FIG. 13B

1300

```
public Object methodEntryOneArg(
  Object classref,
  Object methodref,
  Object instance,
  Object selectedArg)
{
  return null;    // Disables methodExit & reportArg instrumentation
}

public Object methodEntryOneTwoArg(
  Object classref,
  Object methodref,
  Object instance,
  Object arg1,
  Object arg2)
{
  return null;    // Disables methodExit & reportArg instrumentation
}

// Called at normal instrumented method exit,
// unless returned methodEntry context is null.
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  Object result) { }

// Overloaded versions of methodExit for primitive return types.
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  int result) { }        // Covers boolean, byte, char, short, and int
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  float result) { }
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
```

FIG. 13C

1300

```
  long result) { }
public void methodExit(
  Object context,
  Object classref,
  Object methodref,
  double result) { }
public void methodExit(
  Object context,
  Object classref,
  Object methodref) { }

// Called unconditionally at method exception
public void methodException(
  Object context,
  Object classref,
  Object methodref,
  Throwable e) { }

//-------------------
// Argument reporting
//-------------------

// Called after instrumented method entry, once per arg, if
// argument reporting was instrumented.
public void reportArg(
  Object context,
  Object classref,
  Object methodref,
  int argNumber,          // starts at 1
  Object methodArg)       // The actual argument (reference types)
{
}

// Overloaded versions of reportArg for primitive types.
public void reportArg(
  Object context,
  Object classref,
  Object methodref,
  int argNumber,          // starts at 1
  int methodArg)  // Covers boolean, byte, char, short, and int
{
}
```

FIG. 13D

1300

```
    public void reportArg(
      Object context,
      Object classref,
      Object methodref,
      int argNumber,          // starts at 1
      float methodArg)
    {
    }
    public void reportArg(
      Object context,
      Object classref,
      Object methodref,
      int argNumber,          // starts at 1
      long methodArg)
    {
    }
    public void reportArg(
      Object context,
      Object classref,
      Object methodref,
      int argNumber,          // starts at 1
      double methodArg)
    {
    }

}   // class NullExecCallback
```

FIG. 14

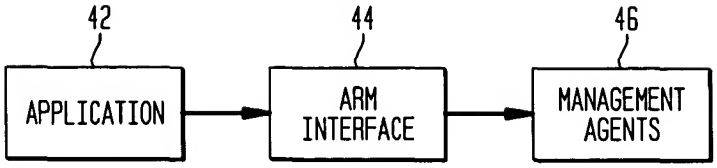


FIG. 15

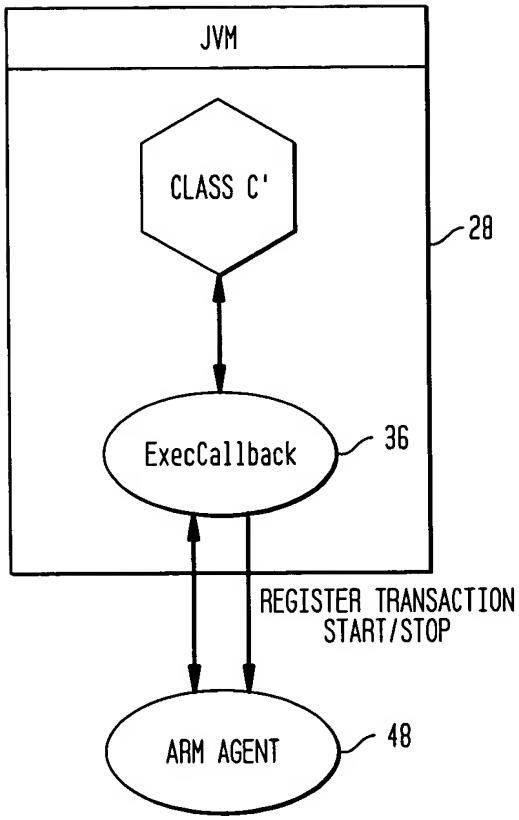

JVM
CLASS C'
28
ExecCallback
36
REGISTER TRANSACTION
START/STOP
ARM AGENT
48

FIG. 16

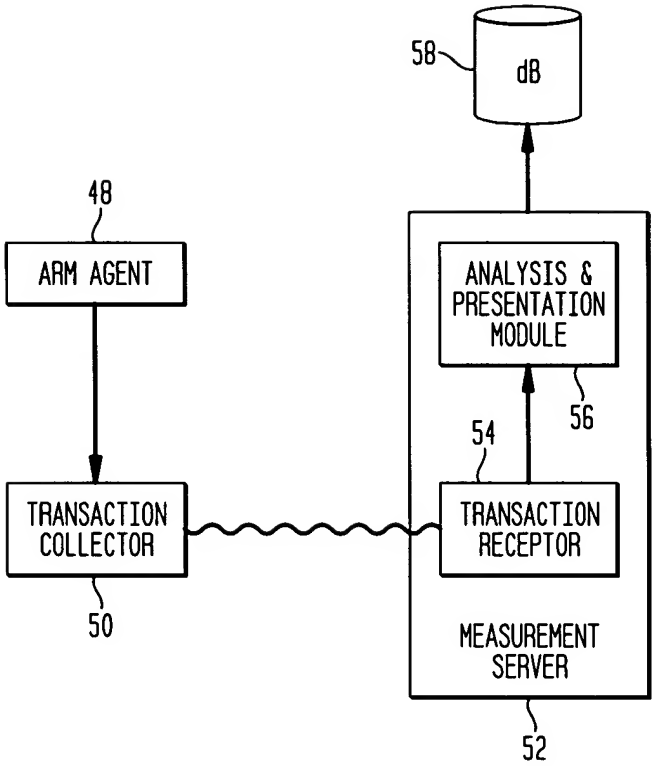

58
dB
48
ARM AGENT
ANALYSIS &
PRESENTATION
MODULE
56
54
TRANSACTION
COLLECTOR
TRANSACTION
RECEPTOR
50
MEASUREMENT
SERVER
52

FIG. 17

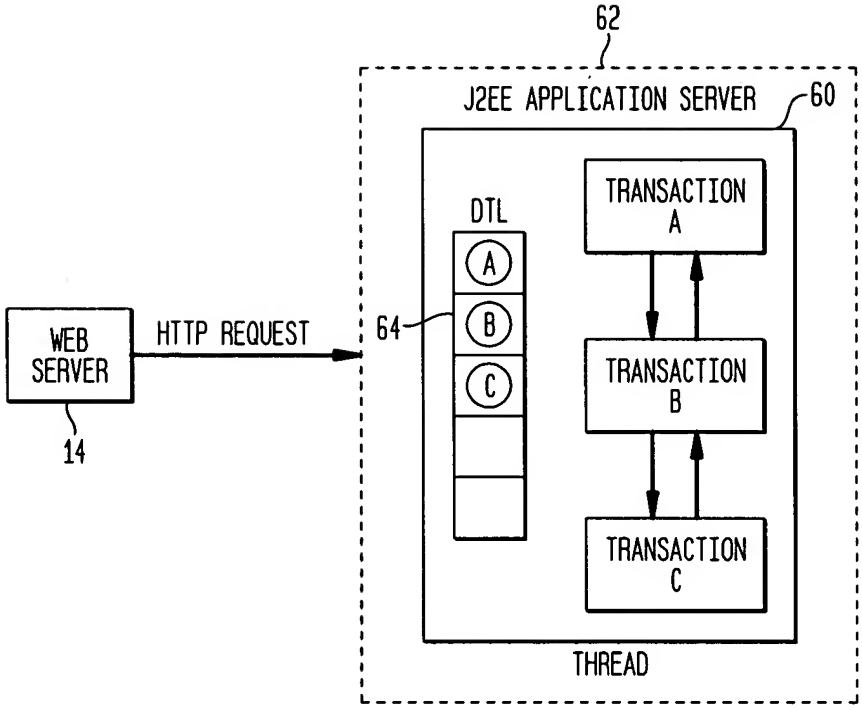

62
J2EE APPLICATION SERVER
60
DTL
A
B
C
64
TRANSACTION A
TRANSACTION B
TRANSACTION C
THREAD
WEB SERVER
14
HTTP REQUEST

FIG. 18

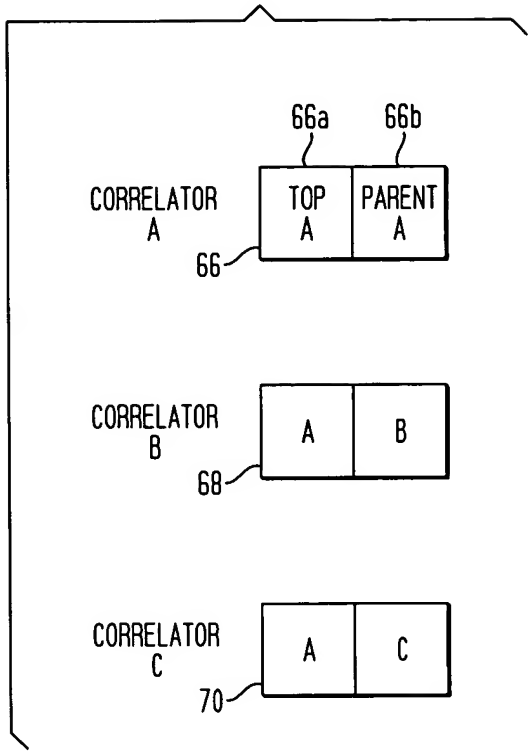

66a
66b
CORRELATOR A
TOP A
PARENT A
66
CORRELATOR B
A
B
68
CORRELATOR C
A
C
70

FIG. 19

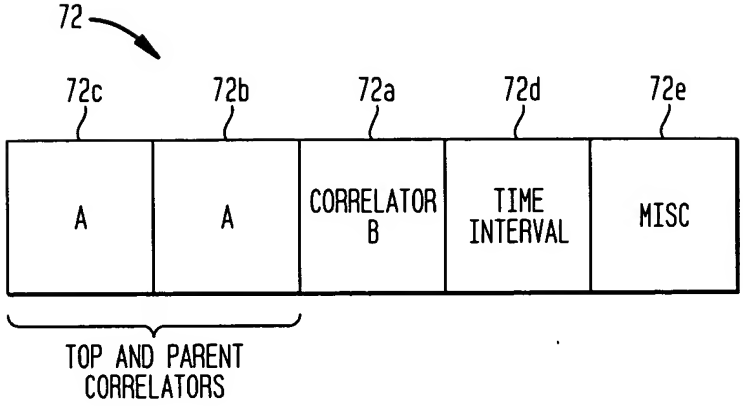

72
72c
72b
72a
72d
72e
A
A
CORRELATOR B
TIME INTERVAL
MISC
TOP AND PARENT CORRELATORS

FIG. 20

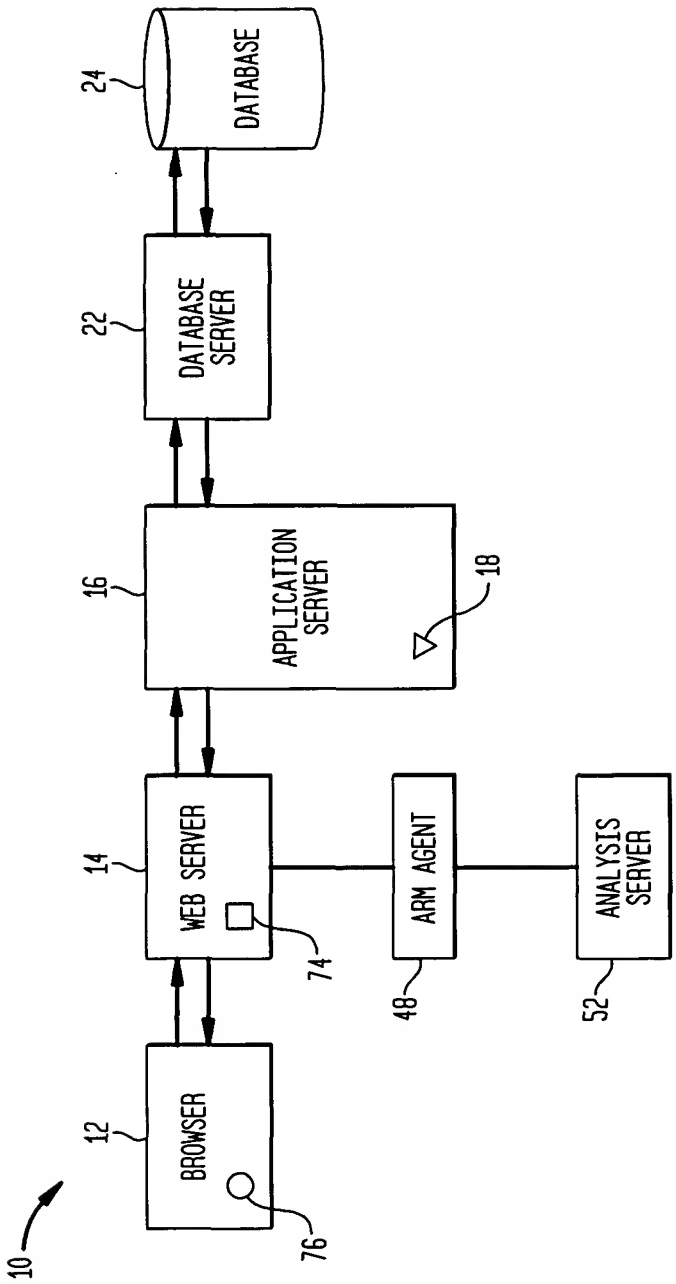

10
12
BROWSER
76
14
WEB SERVER
74
16
APPLICATION SERVER
18
22
DATABASE SERVER
24
DATABASE
48
ARM AGENT
52
ANALYSIS SERVER

FIG. 21

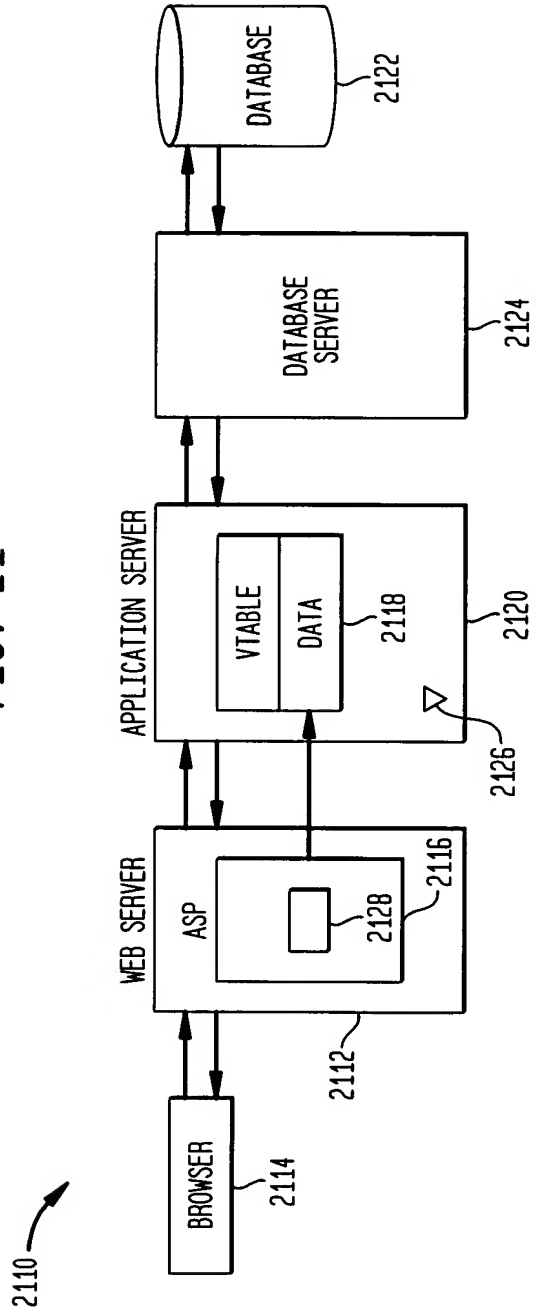

DATABASE
2122
DATABASE SERVER
2124
APPLICATION SERVER
VTABLE
DATA
2118
2120
2126
WEB SERVER
ASP
2128
2116
2112
BROWSER
2114
2110

FIG. 22

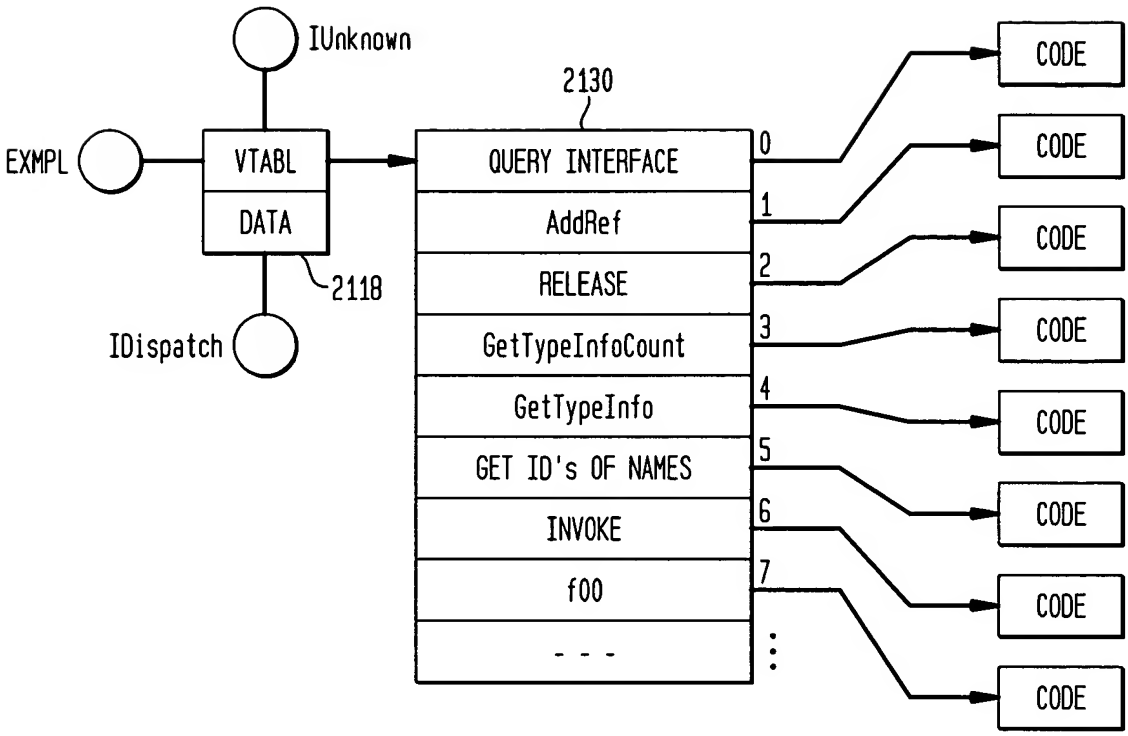

IUnknown
EXMPL
VTABL
DATA
2118
IDispatch
2130
QUERY INTERFACE
AddRef
RELEASE
GetTypeInfoCount
GetTypeInfo
GET ID's OF NAMES
INVOKE
f00
- - -
0
1
2
3
4
5
6
7
CODE
CODE
CODE
CODE
CODE
CODE
CODE
CODE

FIG. 23

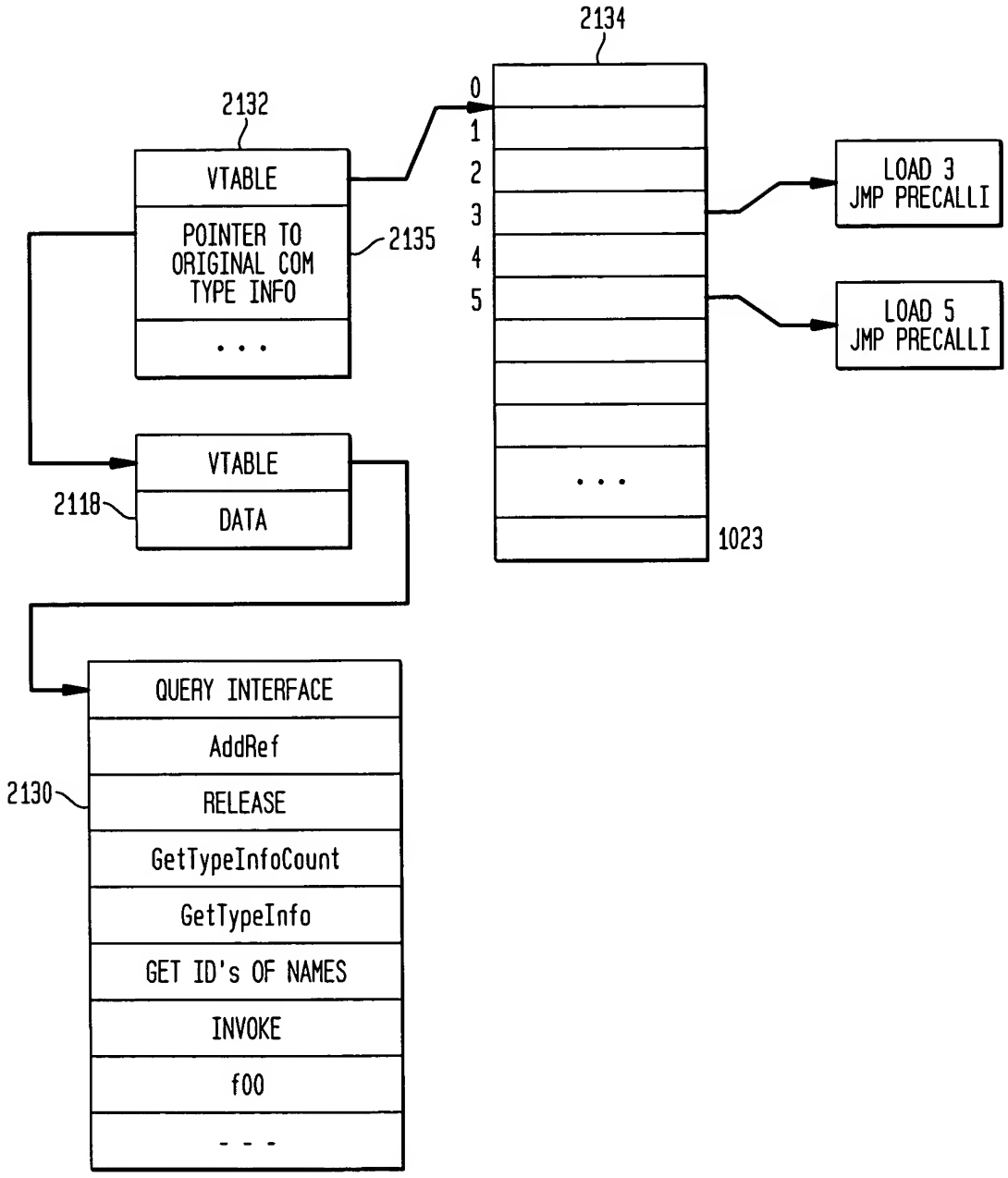

2134
2132
VTABLE
POINTER TO ORIGINAL COM TYPE INFO
2135
. . .
0
1
2
3
4
5
. . .
1023
LOAD 3 JMP PRECALLI
LOAD 5 JMP PRECALLI
VTABLE
DATA
2118
QUERY INTERFACE
AddRef
RELEASE
GetTypeInfoCount
GetTypeInfo
GET ID's OF NAMES
INVOKE
f00
- - -
2130

FIG. 24

```
PrecallInterceptor UNIVERSAL COM METHOD (METHOD #){
        DETERMINE ARGUMENTS NEEDED FOR METHOD #
        ARM START
        CALL ORIGINAL METHOD #
        ARM STOP
        }
```

FIG. 25

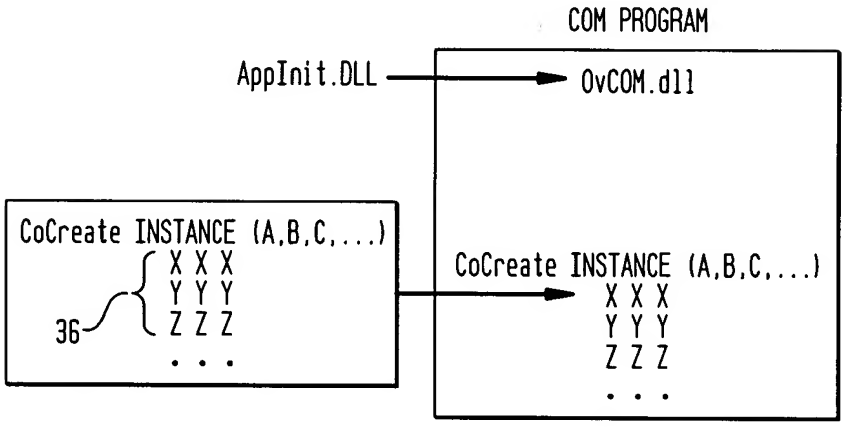


FIG. 26

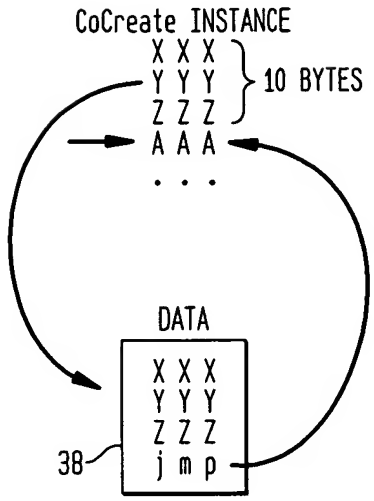


FIG. 27

```
OVTA CoCreateInstance (A,B,C) {
        .
        .
        .
   CALL CoCreateInstance (A,B,C) {
        .
        .
        .
      ACCESS B
      WRAP OBJECT REFERRED BY B
      SET B TO POINT TO WRAPPER OBJECT
      RETURN TO ORIGINAL CALLER
      }
```

FIG. 28

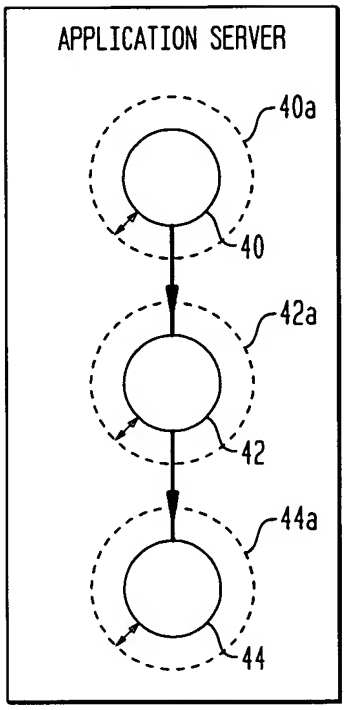
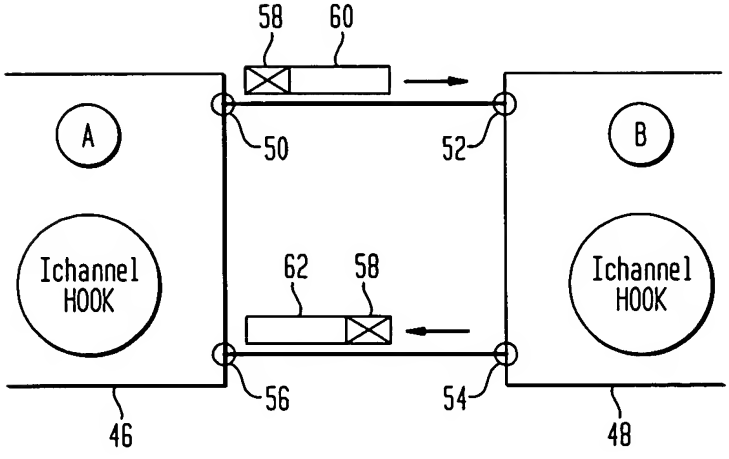

APPLICATION SERVER
40a
40
42a
42
44a
44

FIG. 29


58
60
A
50
52
B
Ichannel HOOK
62
58
Ichannel HOOK
56
54
46
48